Universidad Nacional de Educación a Distancia
E.T.S.I. Informática
Dpto. de Sistemas de Comunicación y Control

PRÁCTICA
DE
Sistemas Distribuidos

(Grado en Ingeniería Informática)

Sistema básico de almacenamiento en la nube usando Java RMI

Curso 2024 - 2025

INTRODUCCIÓN

Este documento contiene el enunciado de la práctica obligatoria de laboratorio correspondiente a la asignatura de Sistemas Distribuidos del Grado de Ingeniería Informática. Lea detenidamente toda la memoria varias veces con el fin de que sus dudas queden resueltas. Si aun así sigue teniendo alguna duda, por favor consúltela a través del foro de la asignatura en el curso virtual.

El software mínimo necesario para la realización de la práctica es el siguiente:

- Kit de desarrollo de Java 11 JDK. Disponible en el servidor de la empresa Oracle, (https://www.oracle.com/java/technologies/downloads/#java11) actual propietaria de la tecnología. En esa dirección pueden encontrar además de la documentación oficial de la API del lenguaje, valiosa información como tutoriales, artículos técnicos, etc.
- Entorno de desarrollo IDE. Aunque no es necesario (se pueden editar los ficheros fuentes con un sencillo editor como el Notepad y compilar/ejecutar desde la línea de comandos), pero sí se recomienda su uso para acelerar el desarrollo. Además, los IDE poseen potentes depuradores que prestan un excelente servicio a la hora de depurar las aplicaciones. Se puede recurrir a cualquier entorno siempre que éste genere código 100% Java. El equipo docente recomienda:

  - Eclipse (http://www.eclipse.org/downloads/)

Les recordamos que la entrega de la práctica es **obligatoria** para todos los estudiantes. Además, es imprescindible aprobar la práctica para aprobar la asignatura. Le recomendamos la lectura detenida de la guía de curso para aclarar cualquier duda sobre los criterios de evaluación.

Una vez entregada la práctica con todo el material requerido, el equipo docente someterá a los programas que se entreguen a un test para comprobar su correcto funcionamiento. Por tanto, se recomienda probar todo el software antes de enviarlo, a ser posible en otra máquina física o virtual diferente, en la que se ha desarrollado.

Si en la convocatoria ordinaria supera la práctica obligatoria pero no supera el examen presencial, se le conservará la calificación de la práctica para la convocatoria extraordinaria del curso actual, pero **nunca** para cursos posteriores. Análogamente sucederá con la nota del examen presencial.

No se conservan calificaciones de ninguna prueba, ya sea teórica o práctica, para cursos posteriores.

Las prácticas se realizan **individualmente**, no se aceptan grupos de trabajo. **La detección de una práctica copiada** de cualquier fuente incluida Internet, obligará al

equipo docente **a ponerlo en conocimiento del Servicio de Inspección de la UNED (CPRI)** para que proceda a la apertura de expediente académico. Para ello, se utilizarán herramientas para detectar este tipo de fraudes como *Turnitin*. Evidentemente, pueden consultar cualquier tipo de dudas, cuestiones, mañas, etc. a través de los foros en el curso virtual de la asignatura.

Lea detalladamente el resto de la memoria ya que se le indica claramente la forma de realizar la práctica, el material que hay que entregar y los plazos para hacerlo. Para cualquier aclaración, no dude en ponerse en contacto con los miembros del equipo docente de la asignatura utilizando los medios que se indican en la Guía del Curso.

ENUNCIADO

El propósito de la práctica es el desarrollo de un software que implemente un sistema de almacenamiento de ficheros en la nube usando Java RMI.

En este sistema actuaran tres tipos de actores/entidades cuyas funciones se listan a continuación:

1.- Servidor: La entidad Servidor se encarga de controlar el proceso de almacenamiento de ficheros y de gestionar los recursos de almacenaje, para ello hace uso de tres servicios:

- Servicio Autenticación: Se encarga de registrar y de autenticar, cuando sea necesario, las otras entidades participantes en el sistema: clientes y repositorios. Ambas entidades se tienen que dar de alta en el sistema y recibir un identificador único para poder operar y realizar almacenaje de ficheros. El registro se lleva a cabo cuando este servicio devuelve a la entidad demandante (cliente o repositorio) el identificador único con el que tiene que autenticarse para cualquier operación que lo requiera en el sistema. El identificador único puede tener el formato que creáis conveniente. Por ejemplo, un número de x cifras que se va incrementando con cada participante nuevo registrado en el sistema.
- Servicio Gestor: Este servicio se encarga de gestionar las operaciones de los clientes en relación con sus ficheros en la nube (físicamente alojados en los repositorios). Usando este servicio, el cliente podrá subir, bajar y borrar ficheros en la nube, además de listar sus ficheros almacenados y compartir ficheros con otros clientes que estén registrados en el sistema.
- Datos: Este servicio hará las funciones de una base de datos que relacione Clientes-Ficheros-Metadatos-Repositorios. Es decir, mantendrá lista de clientes y repositorios conectados al sistema, junto con los ficheros; y los relacionarán permitiendo operaciones de consulta, borrado y añadido. Los dos servicios anteriores (Servicio Autenticación y Servicio Gestor) harán uso de este servicio para realizar las operaciones sobre el estado de las entidades del sistema y sus atributos. Además, también debe registrar el número de veces que se ha descargado un determinado fichero. Aunque podría usarse un sistema de gestión de bases de datos (SGBD) para implementar este servicio, esto haría muy complejo el desarrollo de la práctica y no atendería a los objetivos de la

asignatura. Así pues, **el equipo docente recomienda** para la implementación del servicio las clases *List* y *HashMap* de Java.

2.- Repositorios: Estas entidades son las responsables de guardar en sus dispositivos de almacenamiento los ficheros que los clientes suben a la nube. Cuando un cliente solicita la subida/bajada/borrado de un fichero a través del servicio Gestor del servidor, este servicio Gestor selecciona el repositorio que le corresponde al cliente y le manda al cliente la información necesaria para completar la operación. Para hacer su función, los repositorios hacen públicas las interfaces de sus dos servicios:

- Servicio Cliente-Operador: Este servicio se encarga de las operaciones de subida de ficheros al repositorio y borrado de los mismos. El servicio Gestor del servidor responde a la petición del cliente enviándole la *URL* (Ver apartado recomendaciones al final del texto) de este servicio para que pueda completar su operación.
- Servicio Servidor-Operador: Este servicio tiene un doble objetivo. Por un lado, suministra los métodos necesarios para que el servidor gestione los lugares de almacenamiento para cada cliente y, por otro lado, se encarga de la operación de bajada de ficheros desde el repositorio al cliente, es decir, cuando un cliente quiere bajar un fichero se lo pide al servidor mediante el servicio Gestor. Una vez que el servidor averigua qué repositorio aloja el fichero requerido por el cliente, éste llama a un método del Servicio Servidor-Operador y le pasa la *URL* del cliente para que pueda llamar al método de descarga del servicio DiscoCliente que es el que realmente se encarga de la descarga. De esta manera, no cargamos al servidor con esta operación temporalmente costosa en términos de entrada/salida y conseguimos *Escalabilidad*.

3.- Clientes: Son las entidades que utilizan los usuarios propietarios de los ficheros. Se registran en el sistema a través del servidor para poder subir, gestionar y almacenar sus ficheros en un repositorio en la nube. El cliente publica la interfaz de un servicio cuyo nombre es DiscoCliente, que será utilizado por el servicio Servidor-Operador del repositorio para descargar al disco duro local del cliente el fichero que este considere oportuno.

Recordad, tal y como hemos indicado, que la entidad Servidor **nunca se encarga de subir/bajar los ficheros**, sólo de gestionar estas operaciones y llevar un registro mediante su Servicio Datos. Estas operaciones costosas se harán desde los servicios Servidor-Operador y DiscoCliente para evitar cargar al servidor y permitir la escalabilidad del sistema fácilmente.

Para simplificar, este sistema gestionará los ficheros dentro de una carpeta llamada **ficheros** que se creará (si no existe previamente) al cargar el programa que inicia la entidad repositorio en la ruta que contiene el fichero de arranque de esta entidad. Además, **sólo admitirá la gestión de ficheros** y no la creación de un árbol de carpetas por cada cuenta de cliente en el repositorio, es decir, el cliente podrá subir ficheros, pero no crear carpetas donde guardarlos. **Cada repositorio creará una carpeta** por cada cliente que alojará **todos los ficheros** de este. (Estas carpetas se crearán dentro de la carpeta **ficheros** comentada anteriormente).

Operativa

Inicialmente la entidad Servidor levanta sus tres servicios: Autenticación, Gestor y Datos.

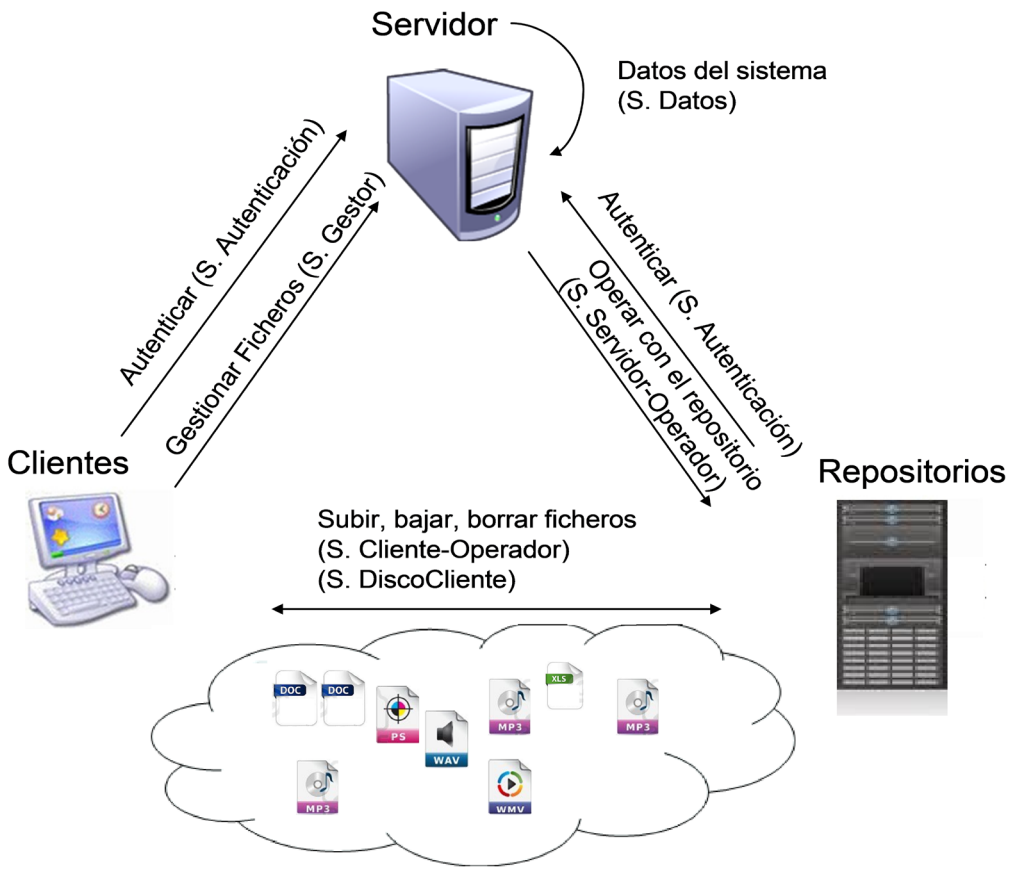
El/Los repositorio/s se autentican en el sistema mediante el servicio Autenticación del servidor, devolviéndole éste un identificador único. El repositorio a partir de este momento está listo para almacenar los ficheros de los clientes.

El/Los cliente/s se autentican en el sistema mediante el servicio Autenticación del servidor, devolviéndole éste un identificador único. En este momento, el servidor le asigna al cliente un repositorio y guarda esta relación a través del servicio Datos. Además, el servidor manda crear una carpeta en el dispositivo de almacenamiento del repositorio mediante el servicio Servidor-Operador. Esta carpeta tendrá por nombre el identificador único del cliente y dentro se alojarán todos sus ficheros en propiedad.

Una vez que el cliente está registrado en el sistema, podrá realizar las operaciones de subida/bajada/borrado de ficheros en la nube. Operaciones gestionadas por el servidor y ejecutadas en la carpeta que cada cliente tiene en el repositorio que le corresponde.

Además, el cliente puede pedir un listado de otros clientes conectados al sistema y permitirles el acceso para la descarga de sus ficheros "Compartir ficheros" **(Está operación de compartir no se exige y será opcional)**.

En la figura siguiente podemos observar de forma esquemática la operativa del sistema:



Interfaz

- El Servidor debe permitir mediante su interfaz las siguientes operaciones:
  1.- Listar Clientes.
  2.- Listar Repositorios.
  3.- Listar Parejas Repositorio-Cliente.
  4.- Salir.

- Los Repositorios deben permitir mediante su interfaz las siguientes operaciones:
  1.- Listar Clientes.
  2.- Listar ficheros del Cliente.
  3.- Salir.

- Los Clientes deben permitir mediante su interfaz las siguientes operaciones:
  1.- Subir fichero.
  2.- Bajar fichero.
  3.- Borrar fichero.
  4.- Número de veces que se ha bajado un fichero.
  5.- Compartir fichero (Opcional).
  6.- Listar ficheros.
  7.- Listar Clientes del sistema.
  8.- Salir.

La opción 4 debe mostrar un listado de ficheros en propiedad del cliente y el número de veces que se han descargado.

En caso de implementarse la opción de compartir ficheros, la operación "6.- Listar ficheros" debe mostrar tanto los ficheros en propiedad del cliente como los que puede obtener a través de la compartición que hacen otros clientes.

IMPORTANTE: Cuando arrancan las aplicaciones de los Clientes y los Repositorios debe aparecer inicialmente un menú con las siguientes opciones, antes de los menús anteriores. Éste debe permitir el registro de un nuevo usuario (cliente o repositorio según corresponda) en el sistema y/o autenticarse:

1.- Registrar un nuevo usuario.
2.- Autenticarse en el sistema (hacer login).
3.- Salir

Especificaciones Generales

- Se debe utilizar código 100% Java JDK. No se admiten librerías de terceros.

- La sintaxis de llamada desde la línea de comandos tiene que ser:
  - servidor (para iniciar el programa Servidor y sus servicios).
  - repositorio (para iniciar un programa Repositorio y sus servicios).
  - cliente (para iniciar un programa Cliente y su servicio).

- Para cumplir con esta sintaxis de llamada hay que crear ficheros en batería (.bat o .sh) para arrancar las aplicaciones. Se recomienda que dentro de estos ficheros haya sólo una línea: **java -jar nombre_fichero.jar**. No incluir dentro de los ficheros en batería órdenes para cambiar el path del sistema operativo, esto no es limpio y no suele funcionar.

- Por motivos de claridad, cada clase Java debe almacenarse en un fichero *.java* Todos los ficheros deben incluir, mediante comentarios al inicio de los mismos, todos los datos del autor: nombre, apellidos y correo electrónico.

- Antes de empezar a programar nada, le recomendamos que se lea varias veces esta memoria y que se haga un pequeño esquema planteando las relaciones entre cada una de las partes funcionales de la aplicación que se le pide.

- Se recomienda encapsular el código dentro de ficheros *.jar*. Esta es una forma elegante, eficiente y compacta de presentar la aplicación final.

- Pueden tomarse las decisiones de diseño que se consideren oportunas, siempre y cuando vayan en consonancia con el enunciado de la práctica y queden perfectamente comentadas en la memoria de la práctica que tiene que entregar.

- En los foros de la asignatura **no se pueden hacer preguntas del tipo**: No me funciona el programa porque me sale este error/excepción. ¿Podrían indicarme dentro de mi código fuente donde está el error o qué funciona mal?

- Aunque se podría utilizar una interfaz gráfica para la práctica, recomendamos la implementación mediante una interfaz en modo texto, ya que no se puntuará en la corrección la implementación con interfaz gráfica y puede ser muy costosa en tiempo.

CONSIDERACIONES DE DESARROLLO

RMI se estudia en el libro de texto básico, aunque es muy recomendable mirar el contenido de los siguientes enlaces:

-http://docs.oracle.com/javase/8/docs/technotes/guides/rmi/index.html, página Web oficial de RMI, donde se proporciona una documentación muy amplia (en inglés) sobre la utilización de RMI y las clases asociadas.

-http://docs.oracle.com/javase/tutorial/rmi/index.html, tutorial que ayuda a entender y comprender la arquitectura de RMI. Es muy recomendable su lectura.

- Los capítulos sobre Java RMI del Libro de la bibliografía complementaria: ISBN: 978-0201796449, Título: Distributed Computing: Principles and Applications., Autor M.L. Liu.,
Editorial Addison Wesley Higher Education. (Existe también versión en **castellano**).

- Sesiones 5, 6 ,7 y 8 de las tutorías Intercampus de este curso.

- Vídeo tutorial del alumno Fermín Silva sobre Java RMI en Eclipse. http://www.youtube.com/watch?v=vnWBrCSjb44

De esta manera, se obtienen los conocimientos mínimos necesarios para ejecutar una aplicación distribuida RMI de ejemplo que sirva de base para el desarrollo de esta práctica.

Por favor, a la hora de implementar vuestra aplicación, **no preguntéis en todas las opciones de los programas ¿Está seguro (s/n)?** como sale en el vídeo de Fermín, ya que es muy farragoso a la hora de corregir.

Además, en el curso virtual de la asignatura, se pondrá un enlace desde donde se podrá **descargar la clase fichero.java**. Esta clase Serializable ya tiene implementados los métodos de lectura y escritura de ficheros en disco, junto con un sistema de comprobación por Checksum.

Detalles sobre las clases de la práctica

Con el objetivo de tener cierto orden, unificación y coherencia en el código que se entrega y para facilitar su posterior corrección, se tienen que nombrar obligatoriamente las clases/interfaces principales del programa de la siguiente manera:

`-Servidor:` Clase que contiene el *main* de la entidad Servidor.
`-Repositorio`: Clase que contiene el *main* de la entidad Repositorio.
`-Cliente`: Clase que contiene el *main* de la entidad Cliente.
`-ServicioAutenticacionInterface`: contiene la interfaz remota del servicio de autenticación que depende de la entidad Servidor.
`-ServicioAutenticacionImpl`: clase que implementa la interfaz remota anterior.
`-ServicioGestorInterface`: contiene la interfaz remota del servicio Gestor que depende de la entidad Servidor.
`-ServicioGestorImpl`: clase que implementa la interfaz remota anterior.
`-ServicioDatosInterface`: contiene la interfaz remota del servicio Datos que depende de la entidad Servidor.
`-ServicioDatosImpl`: clase que implementa la interfaz remota anterior.
`-ServicioSrOperadorInterface`: contiene la interfaz remota del servicio Servidor-Operador que depende de la entidad Repositorio.
`-ServicioSrOperadorImpl`: clase que implementa la interfaz remota anterior.
`-ServicioClOperadorInterface`: contiene la interfaz remota del servicio Cliente-Operador que depende de la entidad Repositorio.
`-ServicioClOperadorImpl`: clase que implementa la interfaz remota anterior.
`-ServicioDiscoClienteInterface`: contiene la interfaz remota del servicio DiscoCliente que depende de la entidad Cliente.
`-ServicioDiscoClienteImpl`: clase que implementa la interfaz remota anterior.
`-Fichero`: clase que implementa las operaciones de lectura/escritura de ficheros en disco (suministrada por el equipo docente).

Aparte de éstas, se pueden utilizar todas las clases que sean necesarias, pero no olvidar describir detalladamente su función en la memoria de la practica, así como, su lugar en el diagrama de clases.

RECOMENDACIONES:

- Nombrar los objetos remotos usando una *URL* que recoja toda la información sobre su dirección-puerto, servicio que presta e identificador de su proveedor; como se hace en el libro de M. L. Liu:

```
URL_nombre="rmi://"+ip+":"+rmiport+"/"+nombre_servicio+"/"+identificador_unico;
Naming.rebind(URL_nombre, objExportado);
```

- Crear una clase metadatos e instanciar un nuevo objeto de este tipo por cada fichero nuevo que se suba al sistema. Esta clase debe tener varios campos que identifique al fichero como su nombre, identificador único de su propietario, etc.

- Desarrollar el código poco a poco e ir compilando conforme se vayan obteniendo unidades funcionales.

- No es necesario cargar *rmiregistry* desde la línea de comandos, es posible y recomendable crear una instancia del mismo mediante:

```
LocateRegistry.createRegistry(port)
```

- No se pide la instalación del gestor de seguridad de Java. Por tanto, **no es necesario instanciar la clase *SecurityManager.***

NORMAS DE ENTREGA

Deberá entregar un único fichero comprimido en formato ZIP que contendrá una carpeta denominada PRACTICA que contendrá los ficheros fuente *.java*, los ejecutables *.class .jar* de la práctica **además de los ficheros *.bat* o *.sh* para arrancar las aplicaciones según la sintaxis de llamada comentada anteriormente**. Este fichero ZIP debe contener en su raíz un fichero PDF que contendrá la memoria de la práctica.

El fichero tiene que llamarse "Nombreestudiante_Apellidosestudiante.zip", donde Nombreestudiante es el nombre del estudiante y Apellidosestudiante son los apellidos del estudiante completos sin usar acentos y usando guiones bajos (_) en vez de espacios en blanco entre nombre y apellidos.

La memoria debe constar de los siguientes apartados:

- Portada con nombre, apellidos, DNI y correo electrónico UNED del estudiante.
- Memoria descriptiva en la que se debe explicar el trabajo realizado. Recurra a esquemas o gráficos para plantear el funcionamiento de los programas y sus funciones.
- Diagramas de clases descriptivas y explicativas de la estructura del sistema.
- Pantallazos de ejemplos de funcionamiento.
- Conclusiones, opiniones, y mejoras (si fuese el caso), relacionadas con la práctica.

El tamaño máximo de este documento memoria es **50 hojas** y el formato *.pdf*

La **no entrega de la memoria supondrá automáticamente un suspenso** en la práctica, aunque esta funcione según lo especificado en este enunciado. La memoria debe ser **original y única**. Cualquier indicio de que no se corresponde con el código que se ha generado o que se ha copiado de otro/a estudiante será tratado como plagio.

La entrega de la práctica se hará a través del curso virtual y los plazos de entrega son:

- Plazo 1 (convocatoria ordinaria): prácticas recibidas antes del 15 de enero.
- Plazo 2 (convocatoria extraordinaria): prácticas recibidas con posterioridad al 15 de enero y antes del 15 de junio.